Real-Time 3D Multi-Person Tracking Using Monte Carlo Surface Sampling

C. Canton-Ferrer, J.R. Casas, M. Pardàs
Image Processing Group - Technical University of Catalonia
Barcelona, Spain

Abstract

The current paper presents a low-complexity approach to the problem of simultaneous tracking of several people in low resolution sequences from multiple calibrated cameras. Redundancy among cameras is exploited to generate a discrete 3D colored representation of the scene. The proposed filtering technique estimates the centroid of a target using only a sparse set of points placed on its surface and making this set evolve along time based on the seminal particle filtering principle. In this case, the likelihood function is based on local neighborhoods computations thus drastically decreasing the computational load of the algorithm. In order to handle multiple interacting targets, a separate filter is assigned to each subject in the scenario while a blocking scheme is employed to model their interactions. Tests over a standard annotated dataset yield quantitative results showing the effectiveness of the proposed technique in both accuracy and real-time performance.

1. Introduction

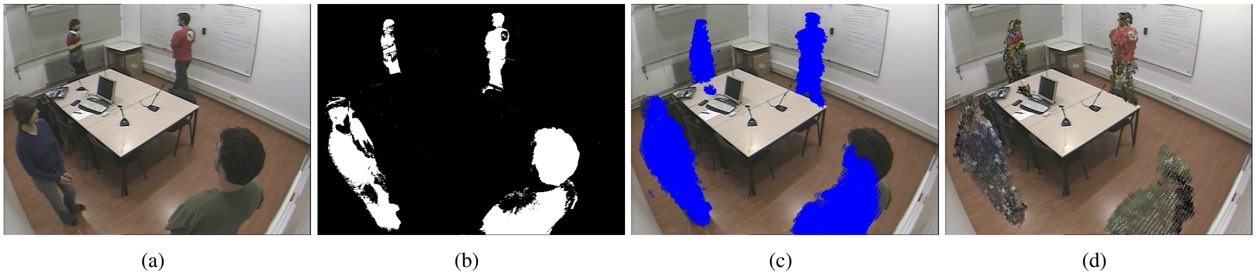
Robust and real-time multi-person tracking algorithms are a key technology required by human-computer interfaces and interaction systems involving assistive or responsive scenarios. This technology provides informative cues such as location and context features commonly employed in a wide range of applications, including gaming, smart environments, surveillance for security and health monitoring. This paper addresses the problem of detecting and tracking a group of people present in a multiple camera setup with a reduced computational footprint.

A number of methods for camera based multi-person 3D tracking has been proposed in the literature [4, 10, 11, 14]. A common goal in these systems is robustness under occlusions created by multiple objects present in the scene when estimating the position of a target. Single camera approaches [14] have been widely employed but are vulnerable to occlusions, rotation and scale changes of the target. In order to avoid these drawbacks, multi-camera tracking techniques [4] exploit spatial redundancy among different views and provide 3D information as well. Integration of features extracted from multiple cameras has been proposed in terms of image correspondences [5], multi-view histograms [11] or voxel reconstructions [7].

Filtering techniques are employed to add temporal consistency to tracks. Kalman filtering approaches have been used to track a single object under Gaussian uncertainty models and linear dynamics [14]. However, these methods do not perform accurately when facing noisy scenes or rapidly maneuvering targets. Monte Carlo based techniques such as particle filtering [2] (PF) have been proposed to cope with these situations since they can deal with multimodal *pdf*s and are able to recover from lost tracks [10].

The proposed algorithm aims at decreasing the computation time by means of a novel tracking technique based on the seminal particle filtering principle. In this case, particles no longer sample the state space but instead a magnitude whose expectancy produces the centroid of the tracked person: the surface voxels. The validity of this claim is supported through Gauss' and Green's theorems applied to the computation of the centroid of a volume exhibiting a radial symmetry on the xy plane. The likelihood evaluation relying on occupancy and color information is computed on local neighborhoods thus dramatically decreasing the computation load of the overall algorithm. Multiple targets are tracked assigning a tracking filter to every one governed by a track management module that classifies whether a blob exhibits an anthropomorphic shape. In order to achieve the most independent set of trackers, we consider a 3D blocking method to model interactions.

Finally, effectiveness of the proposed algorithms is assessed by means of objective performance measures defined in the framework the CLEAR [1] multi-target tracking database. Real-time performance of this algorithm is assessed by presenting the real-time-factor vs. precision and accuracy graphics showing the trade-off between speed and performance. A real-time GPU implementation of this systems shows its adequateness for human-computer interfaces.



(a) (b) (c) (d)

Figure 1. Input data generation example. In (a), a sample of the original images. In (b), foreground segmentation of the input images employed by the SfS algorithm. In (c), example of the binary 3D voxel reconstruction and, in (d), the final colored version shown over a background image.

2. System overview

For a given frame in the video sequence, a set of N images are obtained from the N cameras (see a sample in Fig.1(a)). Each camera is modeled using a pinhole camera model based on perspective projection with camera calibration information available. Foreground regions from input images are obtained using a segmentation algorithm based on Stauffer-Grimson's background learning and subtraction technique [15] as shown in Fig.1(b).

Redundancy among cameras is exploited by means of a Shape-from-Silhouette (SfS) technique [7]. This process generates a discrete binary occupancy representation of the 3D space (voxels) denoted as $\boldsymbol{\mathcal{V}}$ and shown in Fig.1(c). A voxel is labeled as foreground or background by checking the spatial consistency of its projection on the N segmented silhouettes. The visibility of a surface voxel onto a given camera is assessed by computing the discrete ray originating from its optical center to the center of this voxel using Bresenham's algorithm and testing whether this ray intersects with any other foreground voxel. The most saturated color among pixels of the set of cameras that see a surface voxels is assigned to it. An example of this process is depicted in Fig.1(d). Let us denote the set of surface voxels as $\boldsymbol{\mathcal{S}}$ and the set of colored surface voxels as $\boldsymbol{\mathcal{S}}^{\mathrm{C}}$. Since the filtering algorithm that will process these data is robust to inaccuracies and spurious voxels, the criteria employed to select the techniques that go from raw images to the 3D voxel reconstruction are low computational load and the ability of the involved algorithms to be parallelized towards a real-time system performance.

The resulting colored 3D scene reconstruction is fed to a track management module that will create and delete filter instances according to the number of targets in the scene. This module involves a simple naïve Bayes classifier that decides whether a blob in the scene can be considered as a person according to its height, bounding box sizes and overall volume. Information about the environment (dimensions of the room, furniture, etc.) allow discarding tracks that are clearly wrong.

3. 3D position estimation from a random set

Assuming an homogeneous 3D discrete object, its geometric moments can be exactly computed using surface information solely through the discrete Gauss theorem [16] that relates the sum of a function over a discrete, closed surface to a sum of a function over the enclosed discrete region. Other approaches compute these moments exploiting the discrete structure of the representation and performing a layered decomposition and analysis of the object by means of the Green's theorem [8, 13]. In the current scenario, it might be assumed that objects in the scene present a rough radial symmetry in the xy plane, thus simplifying the application the abovementioned divergence theorems and posing the computation of the centroid $\mathbf{x}_t$ as the average of the positions of the surface voxels:

$$\mathbf{x}_t = \frac{\sum\limits_{\mathcal{V}\in\boldsymbol{\mathcal{V}}_t} \mathcal{V}_{\mathbf{x}}}{|\boldsymbol{\mathcal{V}}_t|} = \frac{\sum\limits_{\mathcal{V}\in\boldsymbol{\mathcal{S}}_t} \mathcal{V}_{\mathbf{x}}}{|\boldsymbol{\mathcal{S}}_t|}, \qquad (1)$$

where $|\cdot|$ stands for the cardinality of the enclosed set and $\mathcal{V}_{\mathbf{x}}$ is the position $\mathbf{x} = (x, y, z)$ of voxel $\mathcal{V}$.

However, let us consider the case where only a uniform random spatially selected subset of either $\boldsymbol{\mathcal{V}}$, $\widehat{\boldsymbol{\mathcal{V}}}_t$, or $\boldsymbol{\mathcal{S}}_t$, $\widehat{\boldsymbol{\mathcal{S}}}_t$, is employed in the computation of the centroid,

$$\mathbf{x}_t \approx \tilde{\mathbf{x}}_t = \frac{\sum\limits_{\mathcal{V}\in\widehat{\boldsymbol{\mathcal{V}}}_t} \mathcal{V}_{\mathbf{x}}}{|\widehat{\boldsymbol{\mathcal{V}}}_t|}, \quad \mathbf{x}_t \approx \tilde{\mathbf{x}}_t^S = \frac{\sum\limits_{\mathcal{V}\in\widehat{\boldsymbol{\mathcal{S}}}_t} \mathcal{V}_{\mathbf{x}}}{|\widehat{\boldsymbol{\mathcal{S}}}_t|}. \qquad (2)$$

In this case, the committed estimation error will be inversely proportional to the cardinality of the employed set as shown in Fig.2, proving that it is still possible the achieve a satisfactory estimation precision using either $\widehat{\boldsymbol{\mathcal{V}}}_t$ or $\widehat{\boldsymbol{\mathcal{S}}}_t$.

Let us pose the problem of estimating the centroid of a single object in the scene by analyzing a randomly selected set of voxels from the whole scene, denoted as $\boldsymbol{\mathcal{W}}$. An approach to the computation of the centroid might be:

$$\tilde{\mathbf{x}}_t \approx \frac{\sum\limits_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \rho(\mathcal{W})\mathcal{W}_{\mathbf{x}}}{\sum\limits_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \rho(\mathcal{W})}, \rho(\mathcal{W}) = \left\{ \begin{array}{ll} 1 & \text{if } \mathcal{W} \in \boldsymbol{\mathcal{V}}_t \\ 0 & \text{if } \mathcal{W} \notin \boldsymbol{\mathcal{V}}_t \end{array} \right. , \qquad (3)$$

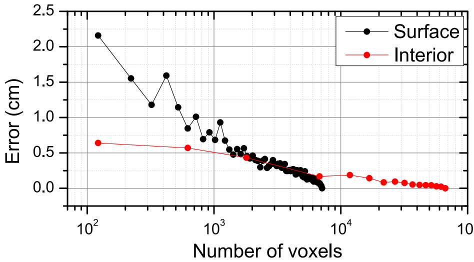


Figure 2. Estimation centroid error when employing a fraction of the surface or interior voxel sets.

where $\rho(\mathcal{W})$ gives the mass density of voxel $\mathcal{W}$. Without a loss of generally, it is assumed that all voxels have the same mass, then $\rho(\cdot)$ is a binary function that checks the occupancy of a given voxel. Therefore, only the fraction of $\boldsymbol{\mathcal{W}}$ that fulfills $\boldsymbol{\mathcal{W}} \in \boldsymbol{\mathcal{V}}_t$ contributes to the computation of $\tilde{\mathbf{x}}_t$. Eq.3 can be rewritten as:

$$\tilde{\mathbf{x}}_t \approx \sum_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \frac{\rho(\mathcal{W})}{\sum\limits_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \rho(\mathcal{W})} \mathcal{W}_{\mathbf{x}} = \sum_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \tilde{\rho}(\mathcal{W}) \mathcal{W}_{\mathbf{x}}, \qquad (4)$$

where $\tilde{\rho}(\mathcal{W}) \in [0,1]$ is the normalized mass contribution of voxel $\mathcal{W}$ in the computation of $\tilde{\mathbf{x}}_t$. In other words, computation of $\tilde{\mathbf{x}}_t$ can be understood as a weighted sum of positions, $\mathcal{W}_{\mathbf{x}}$, where weights $\tilde{\rho}(\mathcal{W})$ assess the likelihood of $\mathcal{W} \in \boldsymbol{\mathcal{V}}_t$. Analogously, this formulation can be extended to employ surface voxels as:

$$\tilde{\mathbf{x}}_t^S \approx \sum_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \frac{\rho_{\mathrm{S}}(\mathcal{W})}{\sum\limits_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \rho_{\mathrm{S}}(\mathcal{W})} \mathcal{W}_{\mathbf{x}} = \sum_{\mathcal{W}\in\boldsymbol{\mathcal{W}}_t} \tilde{\rho}_{\mathrm{S}}(\mathcal{W}) \mathcal{W}_{\mathbf{x}}, \qquad (5)$$

where $\rho_{\mathrm{S}}(\mathcal{W}) \in [0,1]$ weights the contribution of voxel $\mathcal{W}$ into the computation of $\tilde{\mathbf{x}}_t^S$ by measuring the likelihood of $\mathcal{W} \in \boldsymbol{\mathcal{S}}_t$.

Within this context, functions $\rho(\cdot)$ and $\rho_{\mathrm{S}}(\cdot)$ might be understood as likelihood functions and Eq.4 and 5 as a sample based representation of an estimation problem. There is an obvious similarity between this representation and the formulation of PFs but there is a significant difference: while particles in PF represent an instance of the whole state space, our samples ($\mathcal{W} \in \boldsymbol{\mathcal{W}}_t$) are points in the 3D space.

The presented concepts are applied to define the Surface Sampling (SS) algorithm. Let $\mathbf{y}_t^i \in \mathbb{R}^3$, a point in the 3D space and $\omega_t^i \in \mathbb{R}$ its associated weight measuring the likelihood of this position being part of the object or part of its surface[1]. Assuming that these points will be uniformly distributed and the object will exhibit aforementioned symmetry property, the centroid can be computed as:

$$\tilde{\mathbf{x}}_t \approx \sum_{i=1}^{N_{\mathrm{s}}} \omega_t^i \mathbf{y}_t^i, \qquad (6)$$

where N_{s} is the number of sampling points. When using SS we are no longer sampling the state space since $\mathbf{y}_t^i$ can not be considered an instance of the centroid of the target as happened with particles in PF. Hence, we will talk about *samples* instead of *particles* and we will refer to $\{(\mathbf{y}_t^i, \omega_t^i)\}_{i=1}^{N_{\mathrm{s}}}$ as the sampling set. This set will approximate the surface of the k-th target, $\boldsymbol{\mathcal{S}}_k$, and will fulfill the sparsity condition $N_{\mathrm{s}} \ll |\boldsymbol{\mathcal{S}}_k|$.

The main advantage of the SS algorithm is its computational efficiency. While particle likelihoods are computed over all data while sample likelihoods will be computed over a local domain, thus drastically reducing the overall number of operations required.

In order to define a method to recursively estimate $\tilde{\mathbf{x}}_t$ from the sampling set $\{(\mathbf{y}_t^i, \omega_t^i)\}_{i=1}^{N_{\mathrm{s}}}$, a filtering strategy has to be set. Essentially, the proposal is to follow the PF analysis loop (re-sampling, propagation, evaluation and estimation) with some opportune modifications to ensure the convergence of the algorithm.

4. Filter implementation

Two crucial factors are to be considered when implementing the SS algorithm: the sample likelihood evaluation and the sample re-sampling and propagation.

4.1. Sample likelihood evaluation

Associated weight ω_t^i to a sample $\mathbf{y}_t^i$ will measure the likelihood of that 3D position to be part of the surface of the tracked target. Two partial likelihood functions, $p_{\mathrm{Surface}}(\boldsymbol{\mathcal{V}}_t|\, \mathbf{y}_t^i)$ and $p_{\mathrm{Color}}(\boldsymbol{\mathcal{S}}_t^{\mathrm{C}}|\, \mathbf{y}_t^i)$, are linearly combined to form $p(\mathbf{z}_t|\, \mathbf{y}_t^i)$, $\mathbf{z}_t = \{\boldsymbol{\mathcal{V}}_t, \boldsymbol{\mathcal{S}}_t^{\mathrm{C}}\}$, as:

$$p(\mathbf{z}_t|\, \mathbf{y}_t^i) = \lambda\, p_{\mathrm{Surface}}(\boldsymbol{\mathcal{V}}_t|\, \mathbf{y}_t^i) + (1-\lambda)\, p_{\mathrm{Color}}(\boldsymbol{\mathcal{S}}_t^{\mathrm{C}}|\, \mathbf{y}_t^i). \qquad (7)$$

Partial likelihoods will be computed on a local domain centered in the position $\mathbf{y}_t^i$. Let $\mathcal{C}(\mathbf{y}_t^i, q, r)$ be a neighborhood of radius r over a connectivity q domain on the 3D orthogonal grid around a sample place in a voxel position $\mathbf{y}_t^i$. Then, we define the occupancy and color neighborhoods around $\mathbf{y}_t^i$ as $\mathbf{O}_t^i = \boldsymbol{\mathcal{V}}_t \cap \mathcal{C}(\mathbf{y}_t^i, q, r)$ and $\mathbf{C}_t^i = \boldsymbol{\mathcal{S}}_t^{\mathrm{C}} \cap \mathcal{C}(\mathbf{y}_t^i, q, r)$, respectively.

For a given sample i occupying a voxel, its weight associated to the raw data will measure its likelihood to belong to the surface of an object. It can be modeled as:

$$p_{\mathrm{Raw}}(\boldsymbol{\mathcal{V}}_t|\, \mathbf{y}_t^i) = 1 - \left| \frac{2|\mathbf{O}_t^i|}{|\mathcal{C}(\mathbf{y}_t^i, q, r)|} - 1 \right|. \qquad (8)$$

Ideally, when the sample $\mathbf{y}_t^i$ is placed in a surface, half of its associated occupancy neighborhood will be occupied and the other half empty. The proposed expression attains its maximum when this condition is fulfilled. Although this likelihood can be computed using the surface voxel data,

[1] The usage of surface information instead of interior information is discussed in the next section.

$\mathcal{S}_t$, this set tends to be noisy hence not suitable for this computation.

Function $p_{\text{Color}}(\mathcal{S}_t^{\text{C}}|\ \mathbf{y}_t^i)$ can be defined as the likelihood of a sample belonging to the surface corresponding to the k-th target characterized by an adaptive reference color histogram $\mathbf{H}_t^k$:

$$p_{\text{Color}}(\mathcal{S}_t^{\text{C}}|\ \mathbf{y}_t^i) = D(\mathbf{H}_t^k, \mathbf{C}_t^j). \tag{9}$$

Since $\mathbf{C}_t^j$ contains only local color information with reference of the global histogram $\mathbf{H}_t^k$, the distance $D(\cdot)$ is constructed towards giving a measure of the likelihood between this local colored region and $\mathbf{H}_t^k$. For every voxel in $\mathbf{C}_t^j$, it is decided whether it is similar to $\mathbf{H}_t^m$ by selecting the histogram value for the tested color and checking whether it is above a threshold γ. Finally, the ratio between the number of similar color and total voxels in the neighborhood gives the color similarity score. Since reference histogram is updated and changes over time, a variable threshold γ is computed so that the 80% of the values of $\mathbf{H}_t^m$ are taken into account.

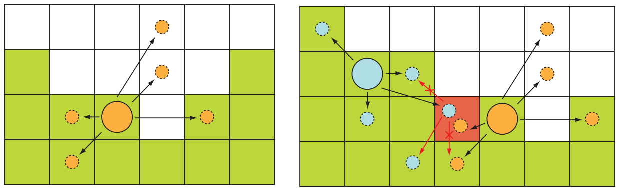
The parameters defining the neighborhood were set to $q = 26$ and $r = 2$ yielding to satisfactory results. Larger values of the radius r did not significantly improve the overall algorithm performance but increased its computational complexity.

Sample propagation and discrete re-sampling

A sample $\mathbf{y}_t^i$ placed near a surface will have an associated weight ω_t^j with a high value. It is a valid assumption to consider that some surrounding positions might also be part of this surface. Hence, placing a number of new samples in the vicinity of $\mathbf{y}_t^j$ might contribute to progressively explore the surface of $\mathcal{S}$.

Given the discrete nature of the 3D voxel space, it will be assumed that every sample is constrained to occupy a single voxel or discrete 3D coordinate and there can not be two samples placed in the same location. Re-sampling method is mimicked from particle filtering so a number of replicas proportional to the normalized weight of the sample are generated. Then, these new samples are propagated and some *discrete* noise is added to their position meaning that their new positions are also constrained to occupy a discrete 3D coordinate (see an example in Fig.3(a)). However, two re-sampled and propagated particles may fall in the same 3D voxel location as shown in Fig.3(b). In such case, one of these particles will randomly explore the adjacent voxels until reaching an empty location; if there is not any suitable location for this particle, it will be dismissed.

The choice of sampling the surface voxels of the object instead of its interior voxels to finally obtain its centroid is motivated by the fact that propagating samples along the surface rapidly spread them all around the object as depicted in Fig.4. Propagating samples on the surface is equivalent to propagate them on a 2D domain, hence the condition of not placing two samples in the same voxel will make them to rapidly explore the surface of the target (see Fig.4(c)). On the other hand, interior voxels propagate on a 3D domain thus having more space to explore and therefore becoming slower to spread (see Fig.4(b)).


(a)

(b)

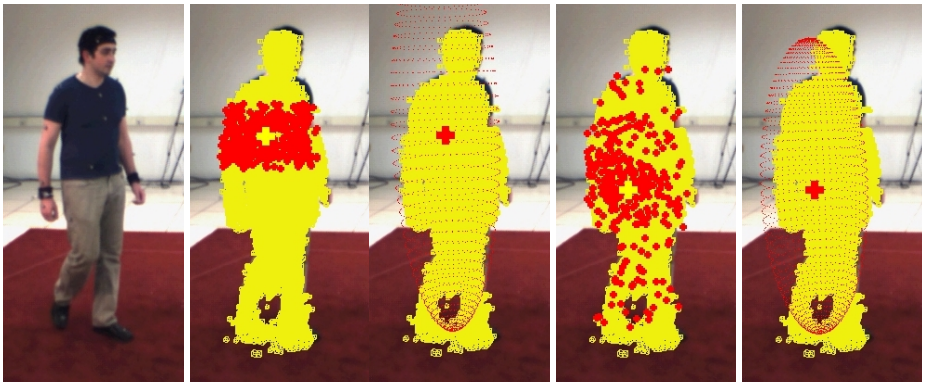
Figure 3. Example of discrete re-sampling and propagation (in 2D). In (a), a sample is re-sampled and its replicas are randomly placed occupying a single voxel. In (b), two re-sampled samples falls in the same position (red cell) and one of them (blue) performs a random search through the adjacent voxels to find an empty location.

Interaction model

The proposed solution for multi-person tracking is to use a split tracker per person together with an interaction model. Let us assume that there are M independent trackers. Nevertheless, they are not fully independent since each tracker can consider voxels from other targets in both the likelihood evaluation or the 3D re-sampling step resulting in target merging or identity mismatches. In order to achieve the most independent set of trackers, we consider a blocking method to model interactions. Many blocking proposals can be found in 2D tracking related works [10] and we extend it to our 3D case. Blocking methods penalize samples that overlap zones with other targets. Hence, blocking information can be also considered when computing the particle weights as:

$$\omega_t^i = p(z_t|y_t^i) \prod_{\substack{k=1 \\ k\neq m}}^{M} \beta\left(\tilde{\mathbf{x}}_{t-1}^m, \tilde{\mathbf{x}}_{t-1}^M\right), \tag{10}$$

where M is the total number of trackers. Term $\beta(\cdot)$ is the blocking function defining exclusion zones that penalize particles that fall into them. For our particular case, considering that people in the room are always sitting or standing up (this is a meeting room so we assume that they never lay down), a way to define an exclusion region modeling the human body is by using an ellipsoid with fixed x and y axis. Axis in z is a function of the estimated centroid height. Tracked objects that come very close can be successfully tracked even though their volumes have partially merged.



(a) Reference image (b) Interior based likelihood (c) Surface based likelihood

Figure 4. Sample positions evolution when using a likelihood based on the interior (a) and surface (b) voxels.

5. Experiments and Results

In order to assess the performance of the proposed tracking systems, they have been tested on the set of benchmarking image sequences provided by the CLEAR Evaluation Campaign 2007 [1]. Typically, these evaluation sequences involved up to 5 people moving around in a meeting room adding up to 5 hours of annotated data. Each sequence was recorded with 4 cameras placed in the corners of a SmartRoom and a zenithal camera placed in the ceiling. All cameras were calibrated and had resolutions ranging from 640x480 to 756x576 pixels at an average frame rate of $f_{\mathrm{R}} = 25$ fps. The test environments was a 5x4 m room with occluding elements such as tables and chairs.

Performance measures proposed in [3] for multi-person tracking evaluation have been adopted. These scores, being used in international evaluation contests [1] and adopted by several research projects such as the European CHIL or the U.S. VACE allow objective and fair comparisons. The two employed performance measures are: the **M**ultiple **O**bject **T**racking **P**recision (*MOTP*), which shows tracker's ability to estimate precise object positions, and the **M**ultiple **O**bject **T**racking **A**ccuracy (*MOTA*), which expresses its performance at estimating the number of objects, and at keeping consistent trajectories. *MOTP* scores the average metric error when estimating multiple target 3D centroids, while *MOTA* evaluates the percentage of frames where targets have been missed, wrongly detected or mismatched. Low *MOTP* and high *MOTA* scores are preferred indicating low metric error when estimating multiple target 3D positions and high tracking performance.

In order to provide a comparison with the widely employed PF approaches to this tracking problem, a standard PF filtering scheme operating over the same input data provided to the SS system is considered. For this PF system, every particle encodes an instance of the target's centroid. An ellipsoid model describing the human shape is employed to compute the likelihood between a particle and the input data $\boldsymbol{\mathcal{V}}$ as the intersection between them. Propagation of particles is driven by a Gaussian drift applied over the state variables of the particle.

Results for the proposed algorithm are depicted in Tab.1 and compared with some other methods that employed the same dataset. However, all these algorithms only reported these performance measures but omitted a quantitative evaluation of its computational requirements. Comparing obtained scores among different algorithms can give an idea about their performance in a scenario where computational complexity is not taken into account. However, if an algorithm requires a high computational load to attain a good performance, this might render it unsuitable for some applications such as instant responsive environments or human-computer interfaces.

Let us define the *RTF* factor of an algorithm as:

$$RTF = \frac{FPS}{f_{\mathrm{R}}}, \tag{11}$$

where *FPS* is the frames-per-second measure of the evaluated algorithm. High values of *RTF* are desirable and $RTF = 1$ is the barrier from real-time to non real-time.

The *RTF* factor associated with a performance measure *MOTP*/*MOTA* (in both vertical axes) of the SS and PF algo-

Method	***MOTP*** (mm)	***MOTA*** (%)
Face detection+Kalman filtering [9]	91	59.66
Appearance models+Particle filtering [12]	141	59.62
Upper body detection+Particle filtering [4]	155	69.58
Zenithal camera analysis+Particle filtering [4]	222	54.94
Voxel analysis+Heuristic tracker [6]	168	30.49
Voxel analysis+Particle filtering (100 particles)	147	74.56
Voxel analysis+Surface sampling (800 samples)	144	81.50

Table 1. Results reported using the CLEAR dataset [1].

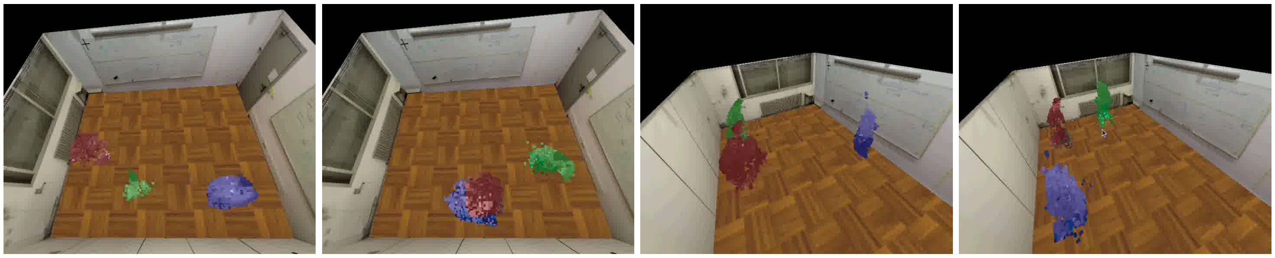


Figure 5. Examples of real-time tracking using the presented SS algorithm in a real scenario using CUDA. The robustness of the algorithm is proved in the cases where two subjects come close to each other or when the data is corrupted or present large missings.

rithms when dealing with raw and colored input voxels is presented in Fig.6. Each point of every curve is the result of an experiment conducted over all the CLEAR dataset associated to a number of samples/particles of each algorithm.

Due to the computational complexity of each algorithm, when comparing SS and PF algorithms under the same operation conditions, the *RTF* associated with SS is always higher than the associated with PF. Similarly, the computational load is higher when analyzing colored than raw inputs. All the plotted curves attain lower *RTF* performance values as the size of the voxel $s_{\mathcal{V}}$ decreases since the amount of data to process increases (note the different *RTF* scale ranges for each voxel size). Regarding the *MOTP*/*MOTA* measures, there is a common tendency to a decrease in the *MOTP* and an increase in the *MOTA* as the *RTF* decreases. The separation between the SS and PF curves is bigger as the voxel size decreases since the PF algorithm has to evaluate a larger amount of data. The observation of these results yield to the conclusion that the SS algorithm is able to produce similar and, in some cases, better results than the PF algorithm with a lower computational cost.

The implementation of these algorithms using 3 PCs equipped with a GPU allowed a real-time implementation of the presented system using CUDA. A voxel size of $s_{\mathcal{V}} = 2$ cm was employed thus producing results usable for realistic human-computer interfaces as shown in Fig.5.

6. Conclusions

This paper presented a multi-person tracking system in a multiple camera view environment intended for real-time operation. Redundant information among cameras is exploited to produce a 3D colored voxel set that is fed to the proposed tracker. A PF based strategy proved efficient for this task but requiring a high computational load. In order to alleviate the high computational load usually required by the widely employed PF filtering strategies, surface sampling technique has been presented as an alternative producing similar results but demanding up to a tenth of the processing time. Color information together with a blocking scheme has been employed to model interactions among targets thus adding robustness against mismatches and cross-overs among targets. Results obtained over a large test database proved the effectiveness of our technique and its implementation using CUDA allowed real-time processing hence rendering this algorithm suitable for human-computer interfaces.

References

[1] CLEAR - Classification of Events, Activities and Relationships Evaluation and Workshop. http://www.clear-evaluation.org, 2007.

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Signal Processing, IEEE Tran. on*, 50(2):174–188, 2002.

[3] K. Bernardin, A. Elbs, and R. Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *IEEE Int. Workshop on Vision Algorithms*, pages 53–68, 2006.

[4] K. Bernardin, T. Gehrig, and R. Stiefelhagen. Multi-level particle filter fusion of features and cues for audio-visual person tracking. In *Proceedings of Classification of Events, Activities and Relationships Evaluation and Workshop*, volume 4625 of *Lecture Notes on Computer Science*, pages 70–81, 2007.

[5] C. Canton-Ferrer, J. R. Casas, and M. Pardàs. Towards a Bayesian approach to robust finding correspondences in multiple view geometry environments. In *Lecture Notes on Computer Science*, volume 3515, pages 281–289, 2005.

[6] C. Canton-Ferrer, J. Salvador, and J. Casas. Multi-person tracking strategies based on voxel analysis. In *Proceedings of Classification of Events, Activities and Relationships Evaluation and Workshop*, volume 4625 of *Lecture Notes on Computer Science*, pages 91–103, 2007.

[7] G. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 714–720, 2000.

[8] J. Crisco and R. McGovern. Efficient calculation of mass moments of intertia for segmented homogeneous three-

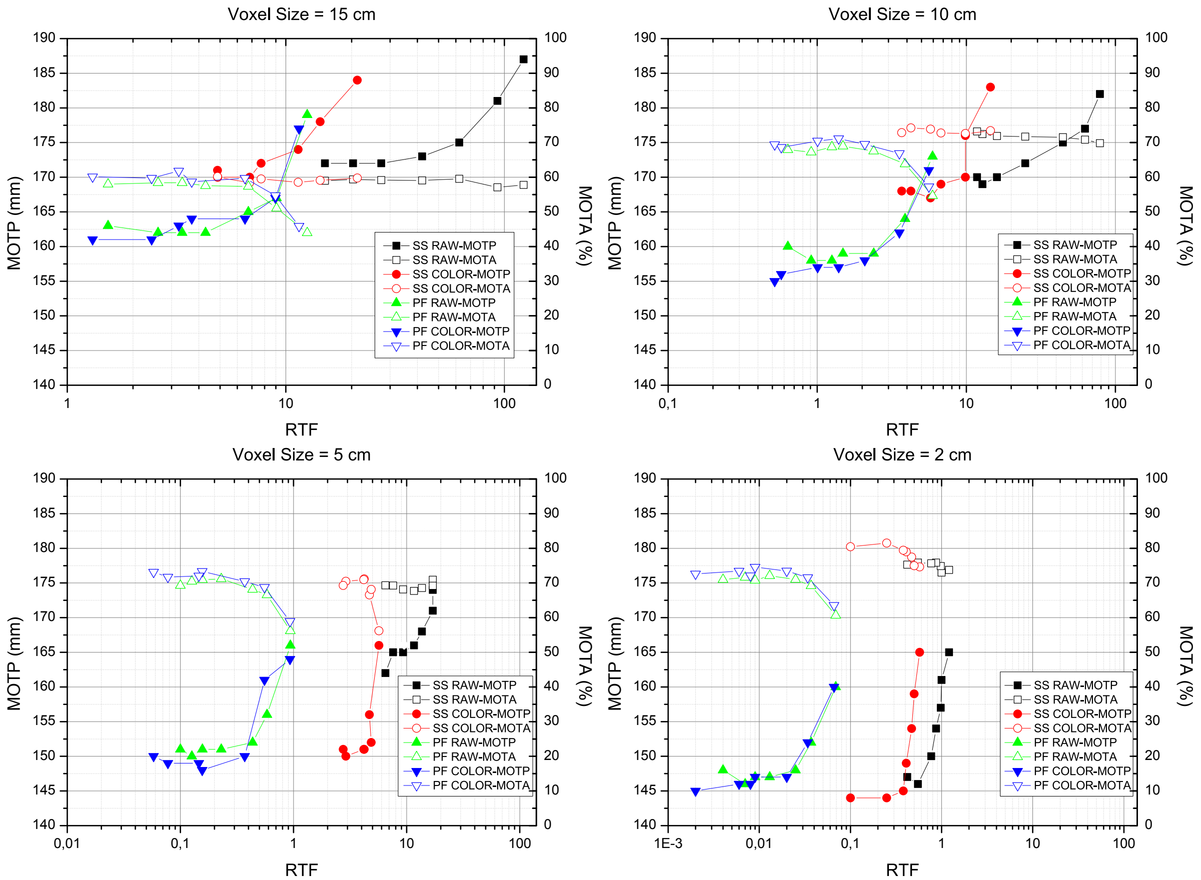


Figure 6. Computational performance comparison among Surface Sampling (SS) and Particle Filtering (PF) using several voxel sizes $s_{\mathcal{V}} = \{2, 5, 10, 15\}$ cm and features (raw or colored voxels). *MOTP* and *MOTA* scores are related to the real-time factor (*RTF*) showing the computational load required by each algorithm to attain a given tracking performance.

dimensional objects. *Journal of Biomechanics*, 31(1):97–101, 1998.

[9] N. Katsarakis, F. Talantzis, A. Pnevmatikakis, and L. Polymenakos. The AIT 3D audio-visual person tracker for CLEAR 2007. In *Proceedings of Classification of Events, Activities and Relationships Evaluation and Workshop*, volume 4625 of *Lecture Notes on Computer Science*, pages 35–46, 2007.

[10] Z. Khan, T. Balch, and F. Dellaert. Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model. In *Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 254–259, 2003.

[11] O. Lanz. Approximate Bayesian multibody tracking. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 28(9):1436–1449, 2006.

[12] O. Lanz, P. Chippendale, and R. Brunelli. An appearance-based particle filter for visual tracking in smart rooms. In *Proceedings of Classification of Events, Activities and Relationships Evaluation and Workshop*, volume 4625 of *Lecture Notes on Computer Science*, pages 57–69, 2007.

[13] J. Leu. Computing a shape's moments from its boundary. *Pattern Recognition*, 24(10):116–122, 1991.

[14] A. Pnevmatikakis and L. Polymenakos. 2D person tracking using Kalman filtering and adaptive background learning in a feedback loop. In *Lecture Notes on Computer Science*, volume 4122, pages 151–160, 2007.

[15] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 252–259, 1999.

[16] L. Yang, F. Albregtsen, and T. Taxt. Fast computation of 3D geometric moments using a discrete Gauss' theorem. *Lecture Notes on Compter Science*, 970:649–654, 1995.